Exhibit 22 to Amended Complaint
Intellectual Ventures I LLC and Intellectual Ventures II LLC

[redacted]

Example Southwest Count IX Systems and Services
U.S. Patent No. 8,352,584 ("the '584 Patent")

The Accused Systems and Services include without limitation Southwest systems and services that utilize Kubernetes; all past, current, and future systems and services that operate in the same or substantially similar manner as the specifically identified systems and services; and all past, current, and future Southwest systems and services that have the same or substantially similar features as the specifically identified systems and services ("Example Southwest Count IX Systems and Services" or "Southwest Systems and Services").[1]

On information and belief, the Southwest Systems and Services use Kubernetes in Southwest's private cloud(s). For example, Southwest posts, or has posted, job opportunities that require familiarity with Kubernetes concepts.

See https://www.linkedin.com/in/madhuker-daraboina-0038001a5/, job profile of Senior DevOps/Cloud Engineer stating use of Kubernetes. (Last accessed on 9/24/2024).

See https://www.linkedin.com/in/abhijitroy18/, job profile of Platform engineer stating use of Kubernetes. (Last accessed on 9/24/2024).

See https://www.linkedin.com/in/hammad--raza/, job profile of senior security engineer stating use of Kubernetes. (Last accessed on 9/24/2024).

See https://www.linkedin.com/in/saikumar-kada-a8b884135/, job profile of senior tech ops Engineer listing Kubernetes as a skill for Southwest position. (Last accessed on 9/24/2024).

[1] For the avoidance of doubt, Plaintiffs do not accuse the public clouds of Defendant, to the extent those services are provided by a cloud provider with a license to Plaintiffs' patents that covers Defendant's activities. [redacted] Further, Plaintiffs do not accuse the public clouds of Defendants if those services are provided by a cloud provider with a license to Plaintiffs' patents that covers Defendants' activities. [redacted] Plaintiffs accuse Defendant private clouds that implement Kubernetes and non-licensed public clouds that Defendant uses to support Kubernetes for its systems and services. [redacted]

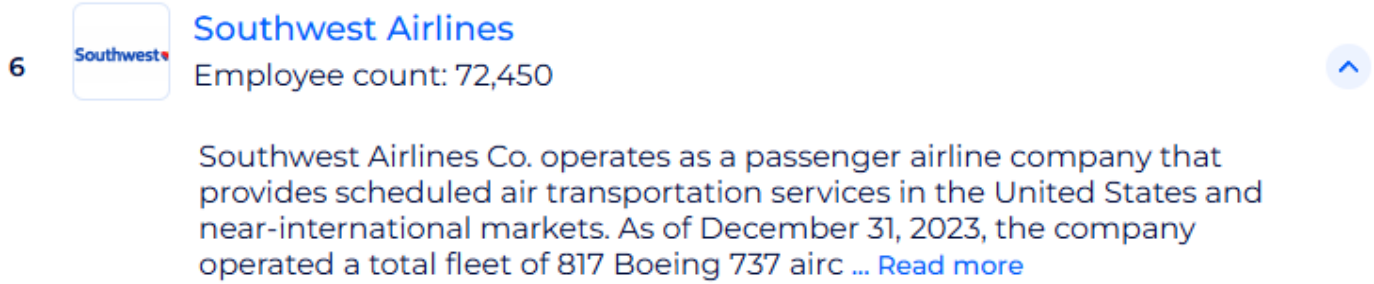
[redacted]

As another example, Southwest has stated that it is investing in cloud technology and has "moved about 50% of its technology" to the cloud and has indicated cloud migration is one of its areas of focus for 2024 and beyond.

Source: https://www.phocuswire.com/southwest-airlines-cio-tech-investment.

On information and belief, other information confirms Southwest uses Kubernetes technology.

[redacted]



Source: https://www.zoominfo.com/tech/23715/kubernetes-tech-from-transportation-airline-industry-by-revenue.[2]

[2] Unless otherwise noted, all sources cited in this document were publicly accessible as of the filing date of the Amended Complaint.

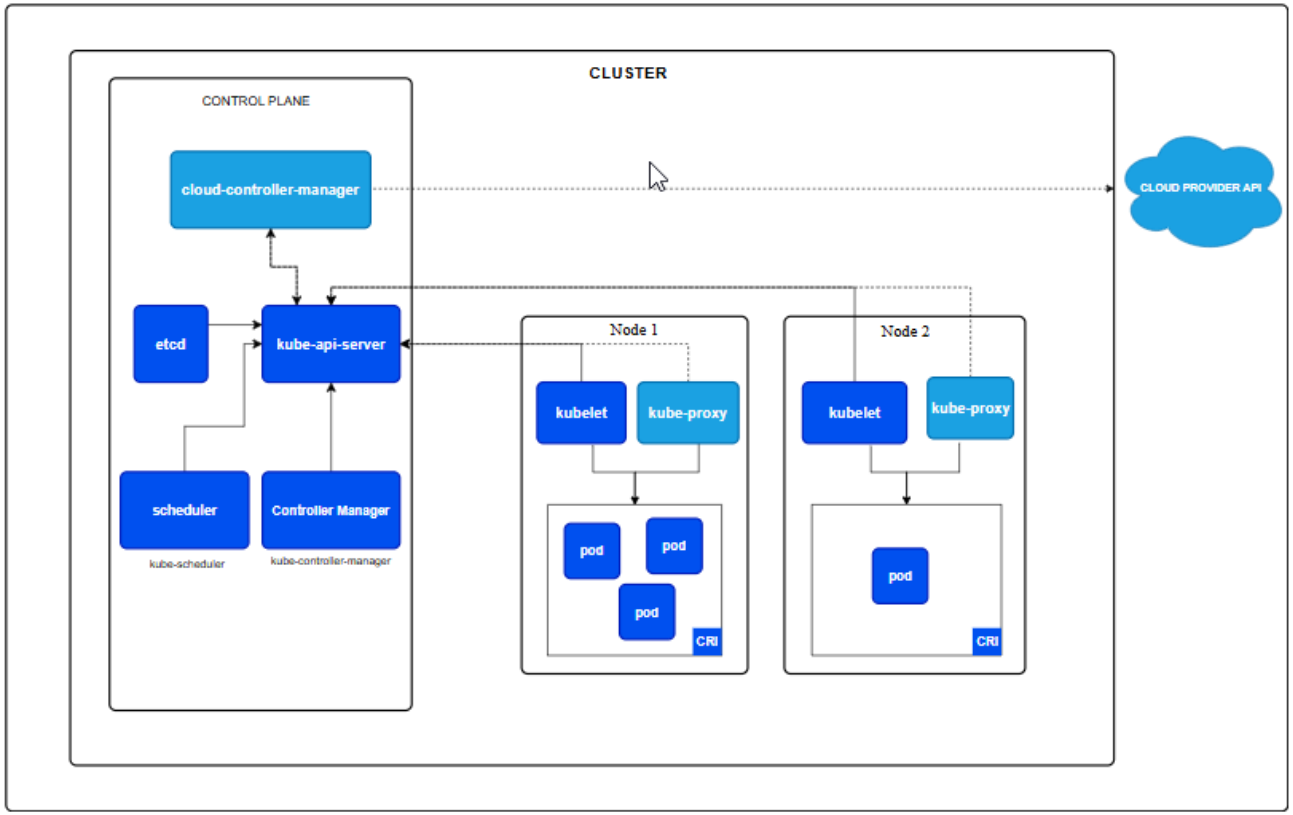
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
[1.pre]. A computer system, comprising:	To the extent this preamble is limiting, on information and belief, the Southwest Count IX Systems and Services include a computer system. Kubernetes hosted on a server system, is a portable, extensible, open-source platform for managing containerized workloads and services. **Overview** Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available. Source: https://kubernetes.io/docs/concepts/overview/. Further, Kubernetes uses clusters, which are groups of nodes that host and run containerized applications as per their defined Deployments and Services.

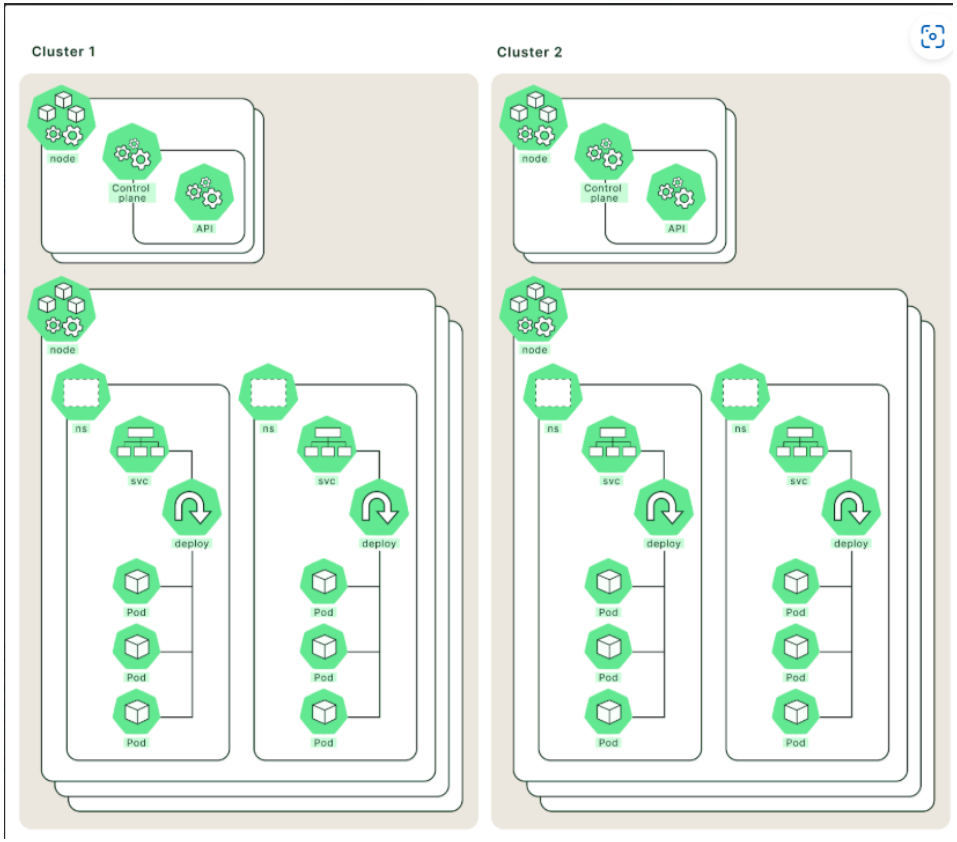
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Cluster Architecture The architectural concepts behind Kubernetes. A Kubernetes cluster consists of a control plane plus a set of worker machines, called nodes, that run containerized applications. Every cluster needs at least one worker node in order to run Pods. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability. Source: https://kubernetes.io/docs/concepts/architecture/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Figure 1. Kubernetes cluster components. Source: https://kubernetes.io/docs/concepts/architecture/. As shown below, Kubernetes supports multiple clusters, which can be hosted on a server system.

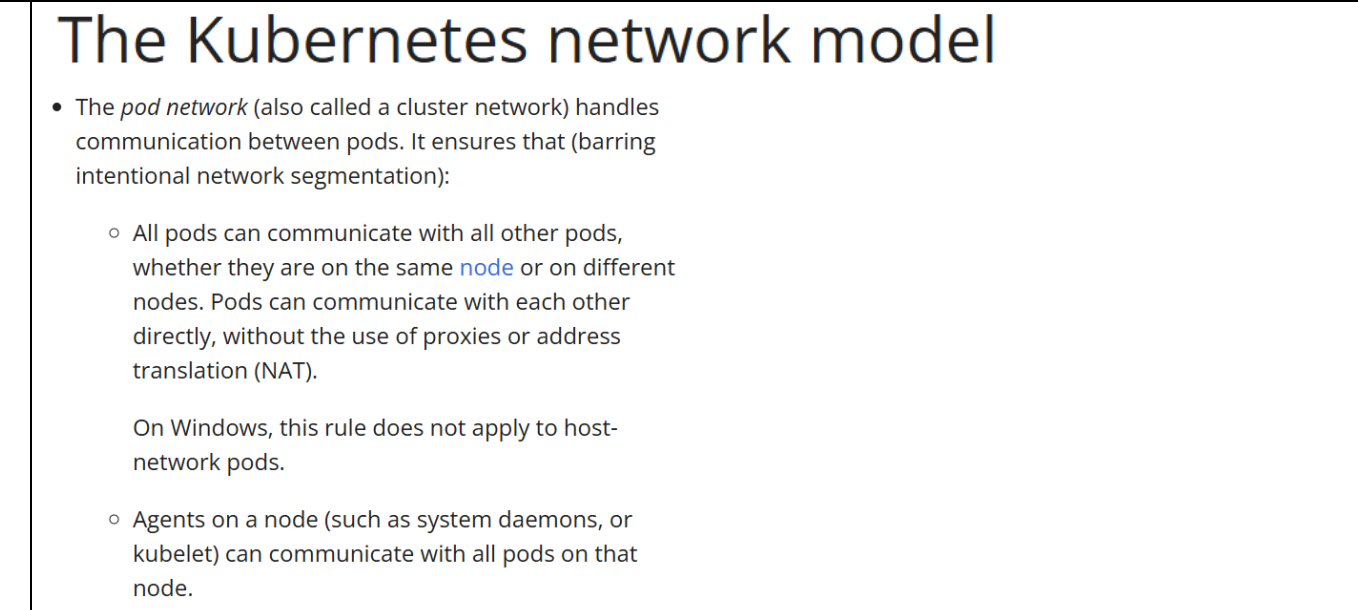
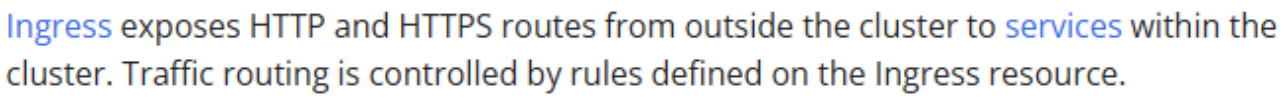
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Kubernetes is designed to scale quickly, massively, and reliably. However, scaling in place — such as adding more nodes to a cluster — eventually reaches a point of diminishing returns. Kubernetes multi-cluster, which is multiple Kubernetes clusters operating as a single logical platform, helps solve this problem and enhances Kubernetes capabilities in many critical areas. Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.  Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.

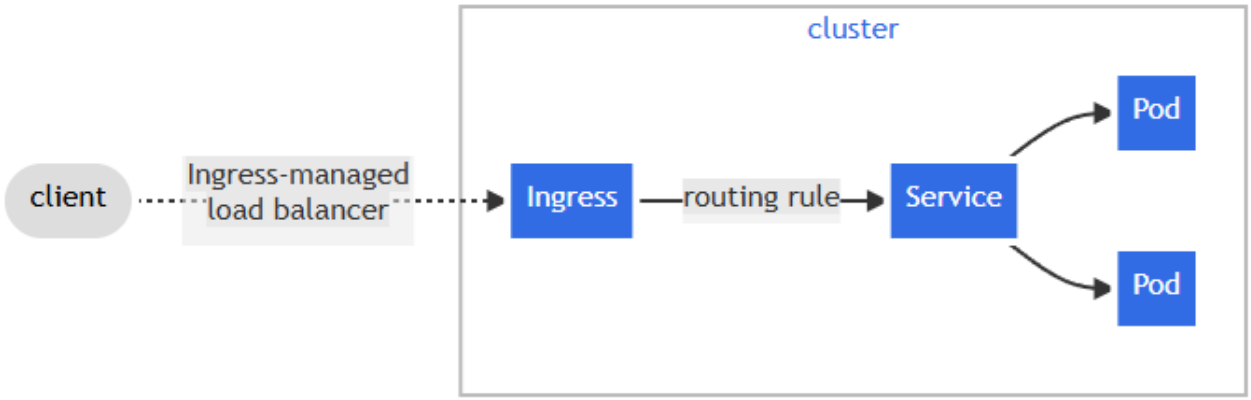
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
[1.a] a private communications network linked to a public communications network;	On information and belief, the Southwest Count IX Systems and Services include a private communications network linked to a public communications network. For example, the Kubernetes system within the Southwest accused network is configured to be linked through, for example, an Ingress, to an external network such as the Internet. For example, Kubernetes supports a Kubernetes cluster network to facilitate communication amongst nodes and/or Pods within the Southwest accused network. Terminology For clarity, this guide defines the following terms: • Node: A worker machine in Kubernetes, part of a cluster. • Cluster: A set of Nodes that run containerized applications managed by Kubernetes. For this example, and in most common Kubernetes deployments, nodes in the cluster are not part of the public internet. • Edge router: A router that enforces the firewall policy for your cluster. This could be a gateway managed by a cloud provider or a physical piece of hardware. • Cluster network: A set of links, logical or physical, that facilitate communication within a cluster according to the Kubernetes networking model. • Service: A Kubernetes Service that identifies a set of Pods using label selectors. Unless mentioned otherwise, Services are assumed to have virtual IPs only routable within the cluster network. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/services-networking/. Further, in Kubernetes, Services provide a way for communication within the Southwest accused network to an external network, such as the Internet. For example, the Ingress is configured to expose HTTP/HTTPS routes from outside the cluster.  Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Figure. Ingress Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Service Expose an application running in your cluster behind a single outward-facing endpoint, even when the workload is split across multiple backends. In Kubernetes, a Service is a method for exposing a network application that is running as one or more Pods in your cluster. A key aim of Services in Kubernetes is that you don't need to modify your existing application to use an unfamiliar service discovery mechanism. You can run code in Pods, whether this is a code designed for a cloud-native world, or an older app you've containerized. You use a Service to make that set of Pods available on the network so that clients can interact with it. Source: https://kubernetes.io/docs/concepts/services-networking/service/. If your workload speaks HTTP, you might choose to use an Ingress to control how web traffic reaches that workload. Ingress is not a Service type, but it acts as the entry point for your cluster. An Ingress lets you consolidate your routing rules into a single resource, so that you can expose multiple components of your workload, running separately in your cluster, behind a single listener. Source: https://kubernetes.io/docs/concepts/services-networking/service/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	A Kubernetes system from within the Southwest accused network includes the Gateway API, which provides Gateway resources. These resources can be defined and specified so that external traffic from the Internet can be routed to various Services. • The Gateway API (or its predecessor, Ingress) allows you to make Services accessible to clients that are outside the cluster. Source: https://kubernetes.io/docs/concepts/services-networking/. Resource model Gateway API has three stable API kinds: • **GatewayClass:** Defines a set of gateways with common configuration and managed by a controller that implements the class. • **Gateway:** Defines an instance of traffic handling infrastructure, such as cloud load balancer. • **HTTPRoute:** Defines HTTP-specific rules for mapping traffic from a Gateway listener to a representation of backend network endpoints. These endpoints are often represented as a Service. Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

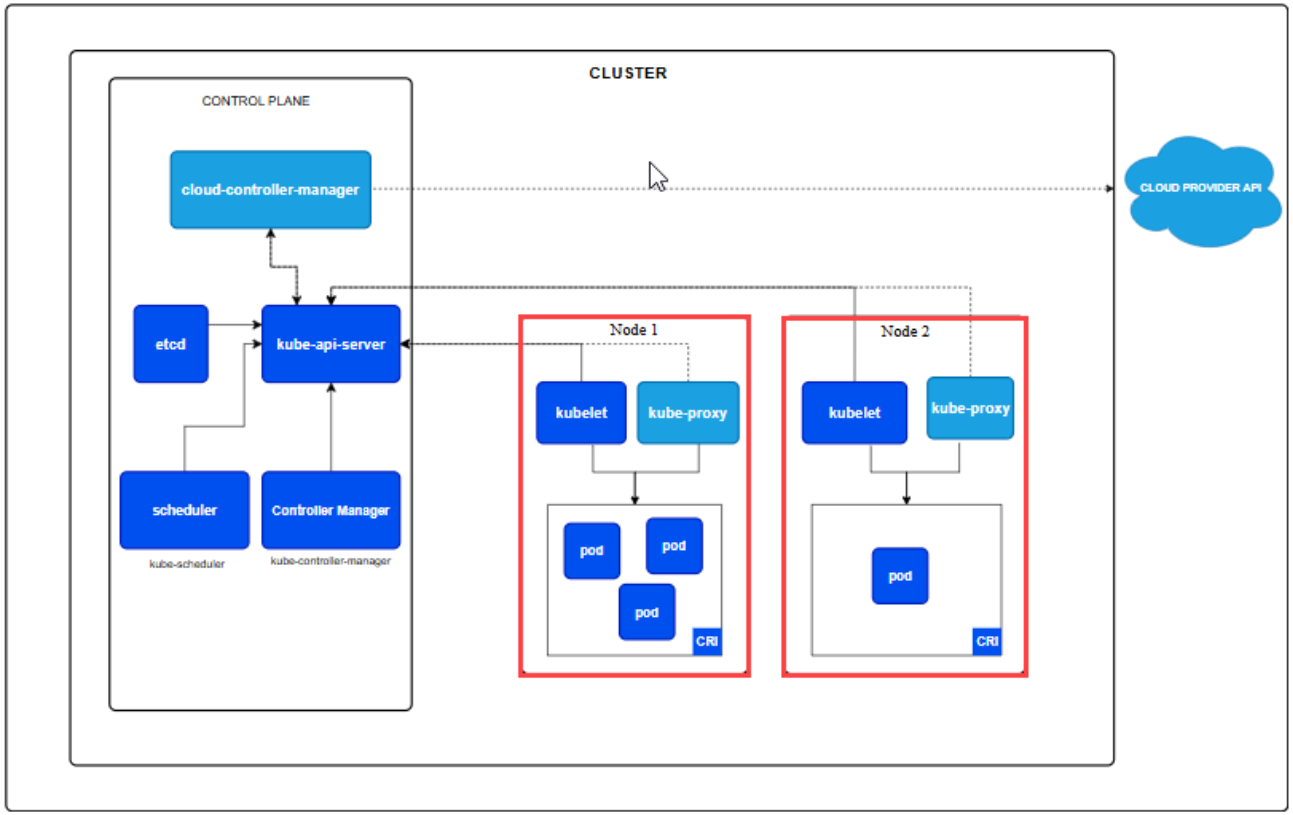
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/services-networking/gateway/. In this example, the request flow for a Gateway implemented as a reverse proxy is: 1. The client starts to prepare an HTTP request for the URL `http://www.example.com` Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.
[1.b] a first cluster comprising a set of computing resources, including at least one hardware processor, in a first configuration, wherein the first cluster is communicatively linked to the private communications network;	On information and belief, the Southwest Count IX Systems and Services include a first cluster comprising a set of computing resources, including at least one hardware processor, in a first configuration, wherein the first cluster is communicatively linked to the private communications network. For example, Kubernetes clusters include at least one node, where a node is either a physical or virtual machine comprising a CPU or portion of CPU resources and memory, to run workloads. These clusters are connected to a private network, for example, the Southwest accused network. On information and belief, the private network, for example, the Southwest accused network(s), facilitates container-to-container, Pod-to-Pod, and/or Pod-to-Services communications across multiple clusters for sharing data and handling tasks.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Cluster Architecture The architectural concepts behind Kubernetes. A Kubernetes cluster consists of a control plane plus a set of worker machines, called nodes, that run containerized applications. Every cluster needs at least one worker node in order to run Pods. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability. Source: https://kubernetes.io/docs/concepts/architecture/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Figure 1. Kubernetes cluster components. Source: https://kubernetes.io/docs/concepts/architecture/.

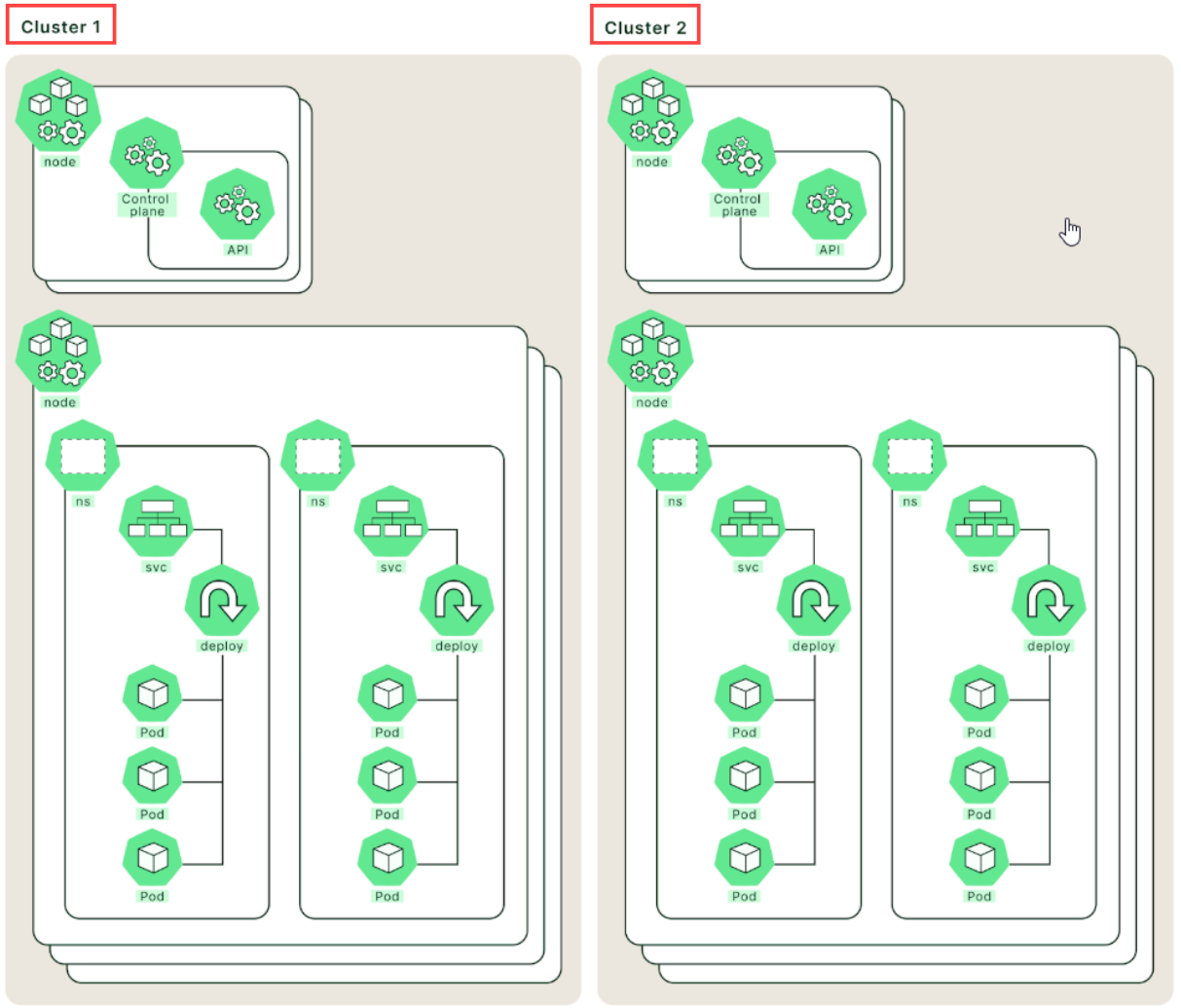
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Networking is a central part of Kubernetes, but it can be challenging to understand exactly how it is expected to work. There are 4 distinct networking problems to address: 1. Highly-coupled container-to-container communications: this is solved by Pods and `localhost` communications. 2. Pod-to-Pod communications: this is the primary focus of this document. 3. Pod-to-Service communications: this is covered by Services. 4. External-to-Service communications: this is also covered by Services. Source: https://kubernetes.io/docs/concepts/cluster-administration/networking/. **Nodes** Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods. Typically you have several nodes in a cluster; in a learning or resource-limited environment, you might have only one node. The components on a node include the kubelet, a container runtime, and the kube-proxy.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Node objects track information about the Node's resource capacity: for example, the amount of memory available and the number of CPUs. Nodes that self register report their capacity during registration. If you manually add a Node, then you need to set the node's capacity information when you add it. Source: https://kubernetes.io/docs/concepts/architecture/nodes/. Kubernetes supports multi-cluster architectures, for example, such as cluster 1 and cluster 2 below, which can be hosted on a server system. Kubernetes is designed to scale quickly, massively, and reliably. However, scaling in place — such as adding more nodes to a cluster — eventually reaches a point of diminishing returns. Kubernetes multi-cluster, which is multiple Kubernetes clusters operating as a single logical platform, helps solve this problem and enhances Kubernetes capabilities in many critical areas. Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.

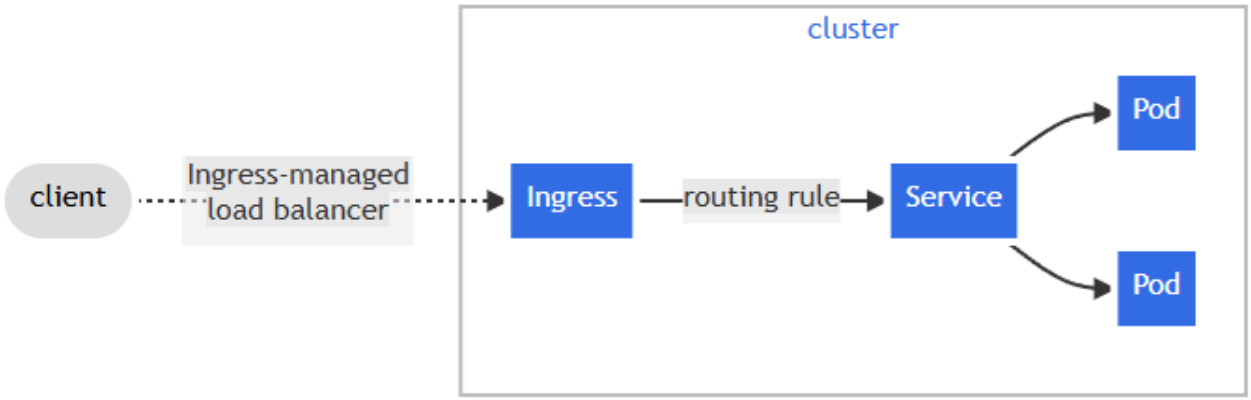
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Kubernetes Multi-Cluster: Why and When To Use Them Application containerization has disrupted the way software applications have been built and deployed. Over the years, Kubernetes has stood out as one of the best platforms for container orchestration. It has helped many companies achieve scalability, resilience, portability, and better resource utilization in their products. However, managing Kubernetes is still complex. The first question which comes to mind is whether we should use a single cluster or a multi-cluster for Kubernetes. Although a single cluster is easy to set up and manage and provides the basic features of Kubernetes, it lacks the typical resilience and high availability Kubernetes is famous for. In many cases, a single cluster is not enough to manage the load efficiently across all components. As a result, we need more than one cluster for a better division of workload and resources, hence the need for a multi-cluster solution. Source: https://www.qovery.com/blog/kubernetes-multi-cluster-why-and-when-to-use-them/. On information and belief, in Kubernetes, Services provide a way for communication within the Southwest accused network to an external network, such as the Internet. For example, the Ingress is configured to expose HTTP/HTTPS routes from outside the cluster. Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Figure. Ingress Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Service Expose an application running in your cluster behind a single outward-facing endpoint, even when the workload is split across multiple backends. In Kubernetes, a Service is a method for exposing a network application that is running as one or more Pods in your cluster. A key aim of Services in Kubernetes is that you don't need to modify your existing application to use an unfamiliar service discovery mechanism. You can run code in Pods, whether this is a code designed for a cloud-native world, or an older app you've containerized. You use a Service to make that set of Pods available on the network so that clients can interact with it. Source: https://kubernetes.io/docs/concepts/services-networking/service/. If your workload speaks HTTP, you might choose to use an Ingress to control how web traffic reaches that workload. Ingress is not a Service type, but it acts as the entry point for your cluster. An Ingress lets you consolidate your routing rules into a single resource, so that you can expose multiple components of your workload, running separately in your cluster, behind a single listener. Source: https://kubernetes.io/docs/concepts/services-networking/service/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	On information and belief, a Kubernetes system from within the Southwest accused network includes the Gateway API, which provides Gateway resources. These resources can be defined and specified so that external traffic from the Internet can be routed to various Services. • The Gateway API (or its predecessor, Ingress) allows you to make Services accessible to clients that are outside the cluster. Source: https://kubernetes.io/docs/concepts/services-networking/. Resource model Gateway API has three stable API kinds: • **GatewayClass:** Defines a set of gateways with common configuration and managed by a controller that implements the class. • **Gateway:** Defines an instance of traffic handling infrastructure, such as cloud load balancer. • **HTTPRoute:** Defines HTTP-specific rules for mapping traffic from a Gateway listener to a representation of backend network endpoints. These endpoints are often represented as a Service. Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/services-networking/gateway/. In this example, the request flow for a Gateway implemented as a reverse proxy is: 1. The client starts to prepare an HTTP request for the URL `http://www.example.com` Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.
[1.c] a second cluster comprising a set of computing resources, including at least one hardware processor, in a second configuration, wherein the second cluster is communicatively linked to the private communications network; and	On information and belief, the Southwest Count IX Systems and Services include a second cluster comprising a set of computing resources, including at least one hardware processor, in a second configuration, wherein the second cluster is communicatively linked to the private communications network. For example, Kubernetes clusters include at least one node, where a node is either a physical or virtual machine comprising a CPU or portion of CPU resources and memory, to run workloads. These clusters are connected to a private network, for example, the Southwest accused network. On information and belief, the private network, for example, the Southwest accused network, facilitates container-to-container, Pod-to-Pod, and/or Pod-to-Services communications across multiple clusters for sharing data and handling tasks.

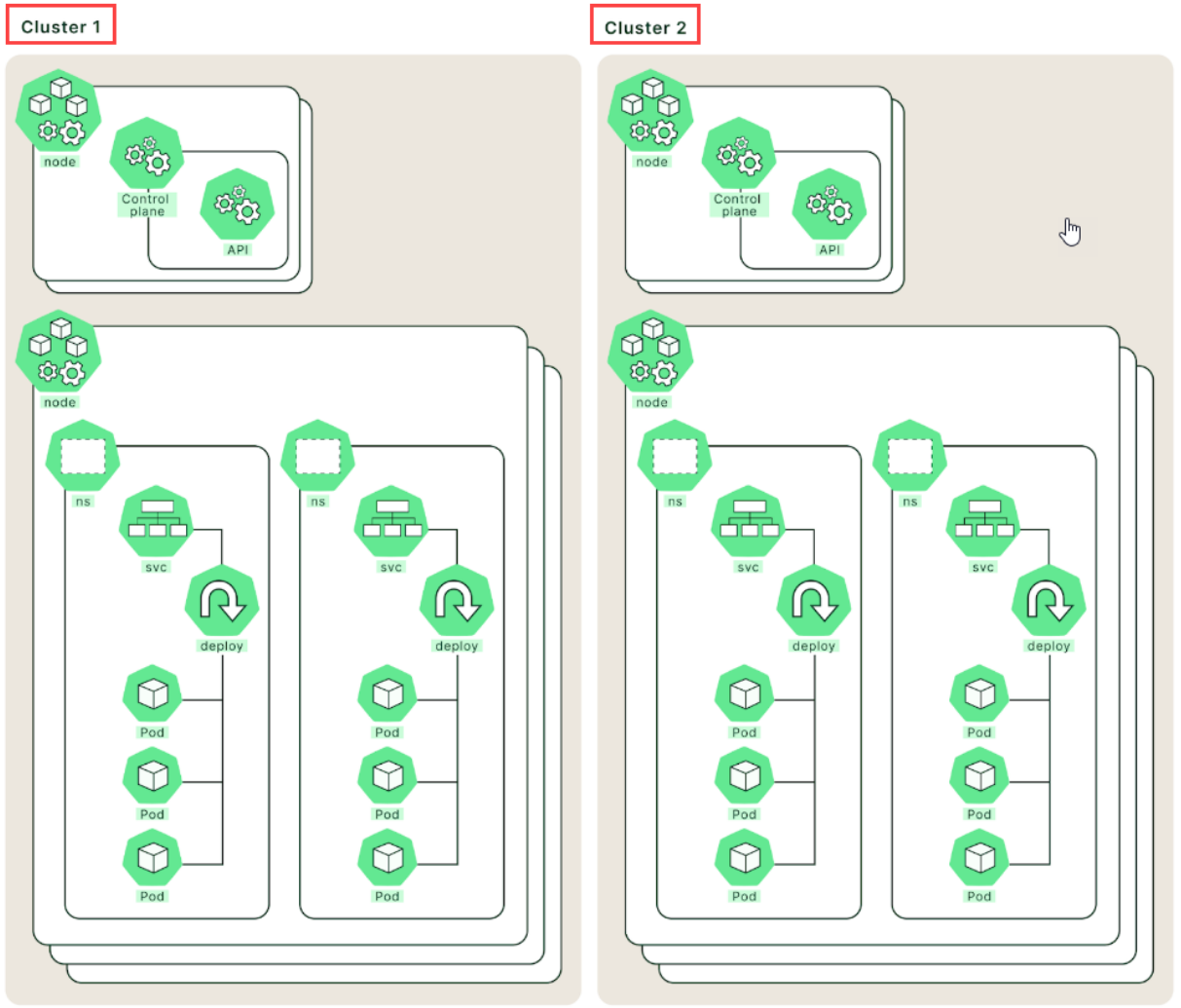
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Cluster Architecture The architectural concepts behind Kubernetes. A Kubernetes cluster consists of a control plane plus a set of worker machines, called nodes, that run containerized applications. Every cluster needs at least one worker node in order to run Pods. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability. Source: https://kubernetes.io/docs/concepts/architecture/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Nodes Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods. Typically you have several nodes in a cluster; in a learning or resource-limited environment, you might have only one node. The components on a node include the kubelet, a container runtime, and the kube-proxy. Node objects track information about the Node's resource capacity: for example, the amount of memory available and the number of CPUs. Nodes that self register report their capacity during registration. If you manually add a Node, then you need to set the node's capacity information when you add it. Source: https://kubernetes.io/docs/concepts/architecture/nodes/. Kubernetes supports multi-cluster architectures, for example, cluster 1 and cluster 2, which can be hosted on a server system. Kubernetes is designed to scale quickly, massively, and reliably. However, scaling in place — such as adding more nodes to a cluster — eventually reaches a point of diminishing returns. Kubernetes multi-cluster, which is multiple Kubernetes clusters operating as a single logical platform, helps solve this problem and enhances Kubernetes capabilities in many critical areas. Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.

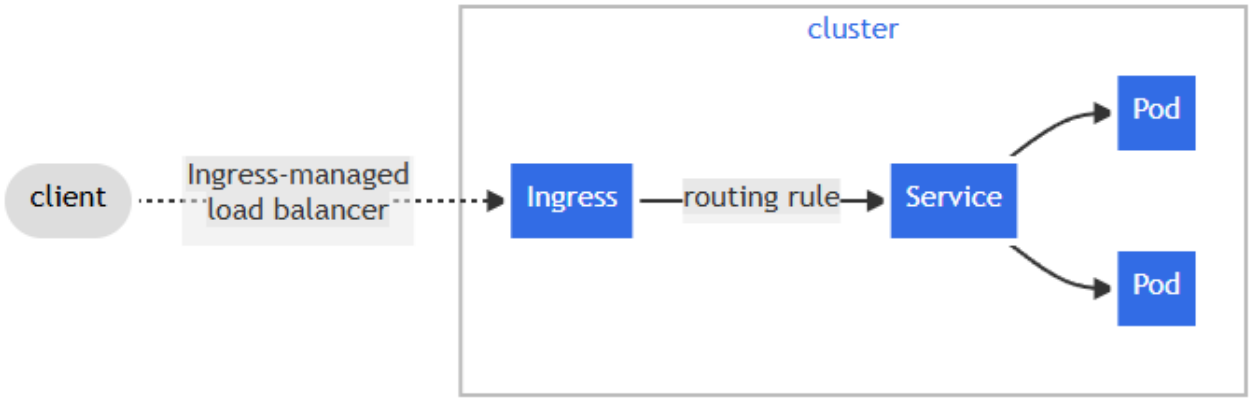
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Kubernetes Multi-Cluster: Why and When To Use Them Application containerization has disrupted the way software applications have been built and deployed. Over the years, Kubernetes has stood out as one of the best platforms for container orchestration. It has helped many companies achieve scalability, resilience, portability, and better resource utilization in their products. However, managing Kubernetes is still complex. The first question which comes to mind is whether we should use a single cluster or a multi-cluster for Kubernetes. Although a single cluster is easy to set up and manage and provides the basic features of Kubernetes, it lacks the typical resilience and high availability Kubernetes is famous for. In many cases, a single cluster is not enough to manage the load efficiently across all components. As a result, we need more than one cluster for a better division of workload and resources, hence the need for a multi-cluster solution. Source: https://www.qovery.com/blog/kubernetes-multi-cluster-why-and-when-to-use-them/. On information and belief, in Kubernetes, Services provide a way for communication within the Southwest accused network to an external network, such as the Internet. For example, the Ingress is configured to expose HTTP/HTTPS routes from outside the cluster. Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Figure. Ingress Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Service Expose an application running in your cluster behind a single outward-facing endpoint, even when the workload is split across multiple backends. In Kubernetes, a Service is a method for exposing a network application that is running as one or more Pods in your cluster. A key aim of Services in Kubernetes is that you don't need to modify your existing application to use an unfamiliar service discovery mechanism. You can run code in Pods, whether this is a code designed for a cloud-native world, or an older app you've containerized. You use a Service to make that set of Pods available on the network so that clients can interact with it. Source: https://kubernetes.io/docs/concepts/services-networking/service/. If your workload speaks HTTP, you might choose to use an Ingress to control how web traffic reaches that workload. Ingress is not a Service type, but it acts as the entry point for your cluster. An Ingress lets you consolidate your routing rules into a single resource, so that you can expose multiple components of your workload, running separately in your cluster, behind a single listener. Source: https://kubernetes.io/docs/concepts/services-networking/service/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	On information and belief, a Kubernetes system from within the Southwest accused network includes the Gateway API, which provides Gateway resources. These resources can be defined and specified so that external traffic from the Internet. • The Gateway API (or its predecessor, Ingress) allows you to make Services accessible to clients that are outside the cluster. Source: https://kubernetes.io/docs/concepts/services-networking/. Resource model Gateway API has three stable API kinds: • **GatewayClass:** Defines a set of gateways with common configuration and managed by a controller that implements the class. • **Gateway:** Defines an instance of traffic handling infrastructure, such as cloud load balancer. • **HTTPRoute:** Defines HTTP-specific rules for mapping traffic from a Gateway listener to a representation of backend network endpoints. These endpoints are often represented as a Service. Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/services-networking/gateway/. In this example, the request flow for a Gateway implemented as a reverse proxy is: 1. The client starts to prepare an HTTP request for the URL `http://www.example.com` Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.
[1.d] a monitoring system to monitor operations of the first cluster and the second cluster for communications problems;	On information and belief, the Southwest Count IX Systems and Services include a monitoring system to monitor operations of the first cluster and the second cluster for communications problems. For example, based on information and belief, Kubernetes supports the Kubernetes API to manage and monitor Kubernetes clusters, for metrics such as cluster health, performance, and/or communication issues. Furthermore, Kubernetes works with full metrics pipeline and monitoring solutions that provide monitoring of operations within Kubernetes clusters, including connectivity problems.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	A full metrics pipeline gives you access to richer metrics. Kubernetes can respond to these metrics by automatically scaling or adapting the cluster based on its current state, using mechanisms such as the Horizontal Pod Autoscaler. The monitoring pipeline fetches metrics from the kubelet and then exposes them to Kubernetes via an adapter by implementing either the `custom.metrics.k8s.io` or `external.metrics.k8s.io` API. Kubernetes is designed to work with OpenMetrics, which is one of the CNCF Observability and Analysis - Monitoring Projects, built upon and carefully extending Prometheus exposition format in almost 100% backwards-compatible ways. Source: https://kubernetes.io/docs/tasks/debug/debug-cluster/resource-usage-monitoring/. **What is Kubernetes monitoring?** Simply put, Kubernetes monitoring is the practice of tracking the status of all components of a Kubernetes environment. Because there are many pieces inside Kubernetes, Kubernetes monitoring actually entails monitoring many distinct things, such as:

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	• The kube-system workloads. • Cluster information using the Kubernetes API. • Applications interactions with Kubernetes by monitoring apps bottom-up. Source: https://www.groundcover.com/kubernetes-monitoring. Kubernetes monitoring includes providing alerts to problems occurring within a cluster, which enables troubleshooting and addressing other potential issues. By collecting Kubernetes data, you'll get viable information regarding your Kubernetes cluster health, that can help you perform Kubernetes troubleshooting and manage issues like unexpected container termination. You can also leverage the data for proactive decisions such as adjusting rate limits. Source: https://www.groundcover.com/kubernetes-monitoring. Furthermore, Kubernetes includes various metrics tracking tools, such as tools for monitoring node health across multiple clusters.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Monitor Node Health *Node Problem Detector* is a daemon for monitoring and reporting about a node's health. You can run Node Problem Detector as a `DaemonSet` or as a standalone daemon. Node Problem Detector collects information about node problems from various daemons and reports these conditions to the API server as Node Conditions or as Events. Source: https://kubernetes.io/docs/tasks/debug/debug-cluster/monitor-node-health/. Kubernetes alerting provides for the identification of operational and connectivity problems, including the identification of clusters associated with the problem.

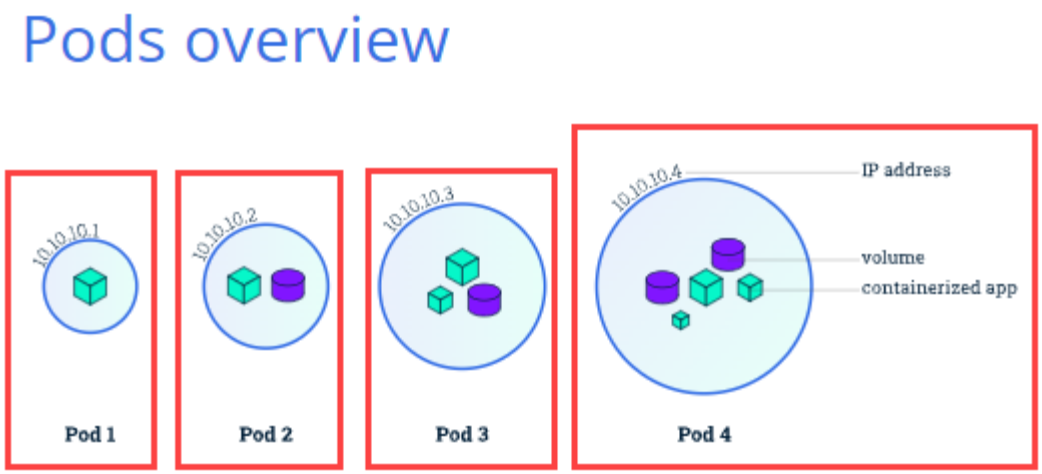
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Kubernetes alerting is the practice of generating notifications for events or trends in Kubernetes that require admins' attention. Examples of such events and trends include: • A node that has failed. • A Pod that is stuck in the pending state. • A container or Pod that is consuming a high level of resources relative to normal consumption trends. • High latency rates for communication between Kubernetes cluster components (such as between kubelet and control plane nodes). Source: https://www.groundcover.com/kubernetes-monitoring/kubernetes-alerting.
[1.e] wherein the first configuration differs from the second configuration;	On information and belief, the Southwest Count IX Systems and Services include a system where the first configuration differs from the second configuration. For example, Kubernetes pods distributed across different nodes in different clusters can be configured to perform different tasks. Each node contains services necessary to run a pod, and each pod runs its own instance of a given application container or set of application containers, resulting in different configurations occurring at each node and cluster.

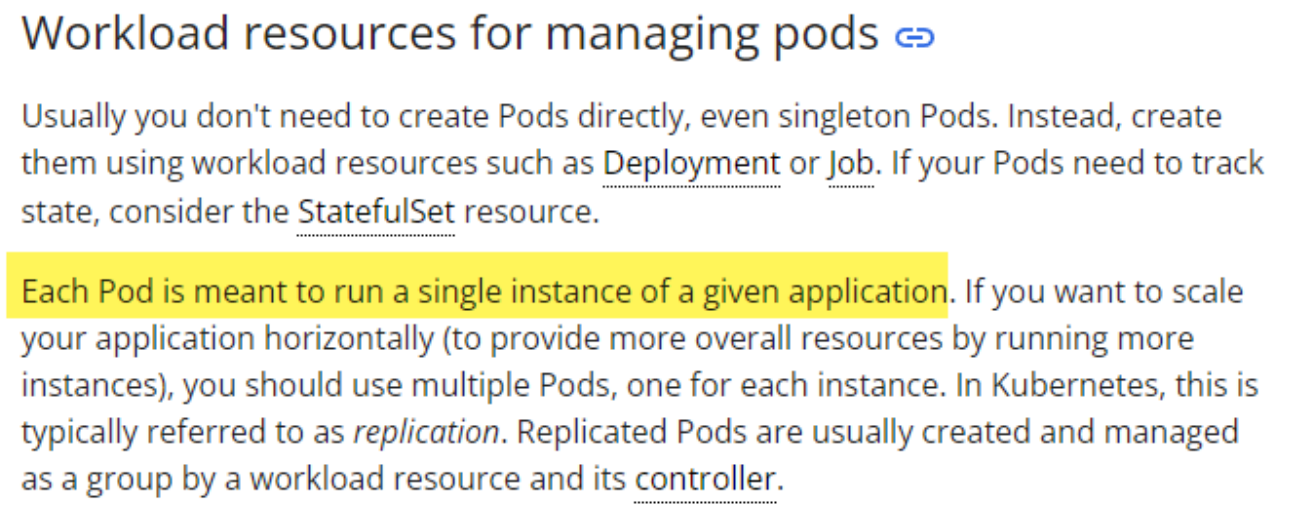
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Pods *Pods* are the smallest deployable units of computing that you can create and manage in Kubernetes. A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host. Source: https://kubernetes.io/docs/concepts/workloads/pods/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/.

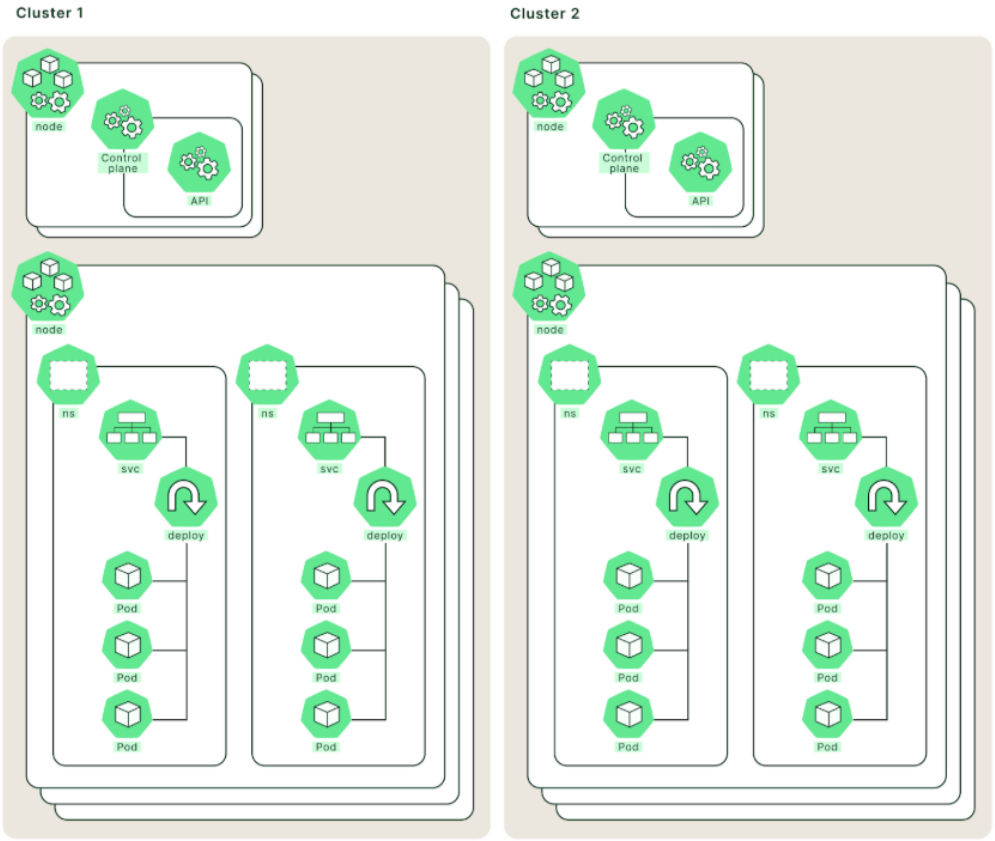
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/workloads/pods/. For example, components of the first cluster architecture differ from the second cluster architecture based on worker node configurations, the running containerized applications within Pods, networking, and other features part of Kubernetes.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Cluster Architecture The architectural concepts behind Kubernetes. A Kubernetes cluster consists of a control plane plus a set of worker machines, called nodes, that run containerized applications. Every cluster needs at least one worker node in order to run Pods. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability. Source: https://kubernetes.io/docs/concepts/architecture/.

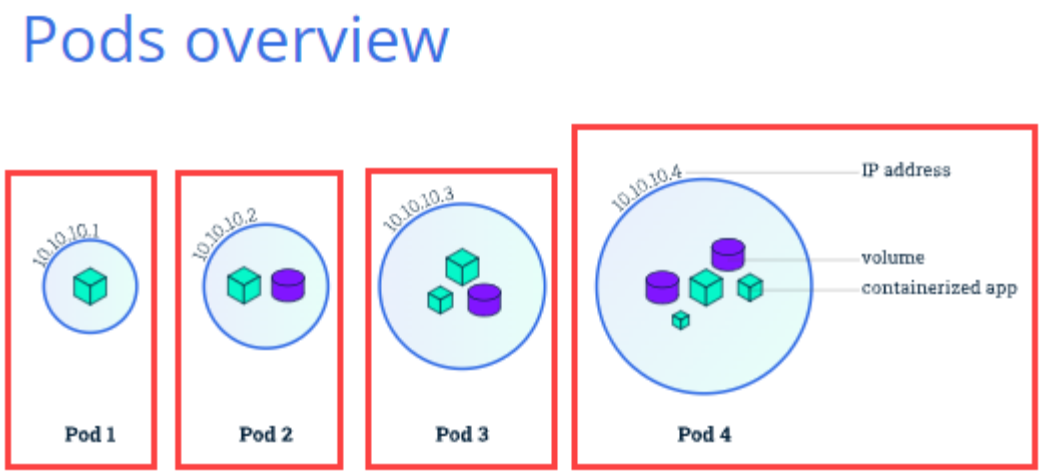
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/.
[1.f] wherein the first configuration provides a first computing environment to perform a first client task and the second configuration	On information and belief, the Southwest Count IX Systems and Services include a system where the first configuration provides a first computing environment to perform a first client task and the second configuration provides a second computing environment to perform a second client task.

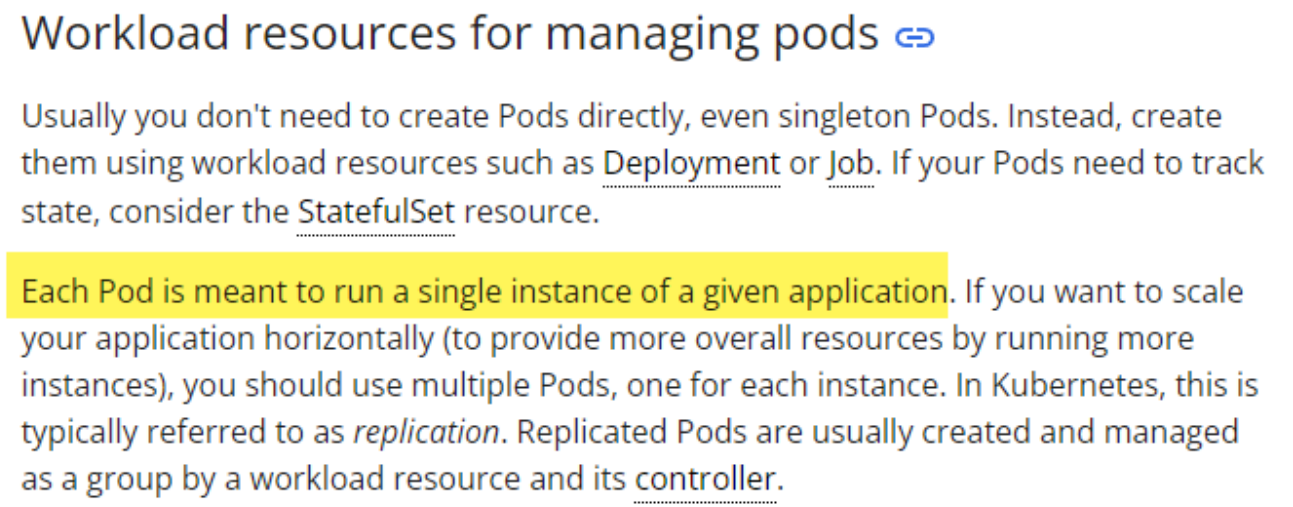
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
provides a second computing environment to perform a second client task; and	For example, Kubernetes pods distributed across different nodes in different clusters can be configured to perform different tasks. Each node contains services necessary to run a pod, and each pod runs its own instance of a given application container or set of application containers, resulting in different configurations occurring at each node and cluster. **Pods** *Pods* are the smallest deployable units of computing that you can create and manage in Kubernetes. A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host. Source: https://kubernetes.io/docs/concepts/workloads/pods/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/.

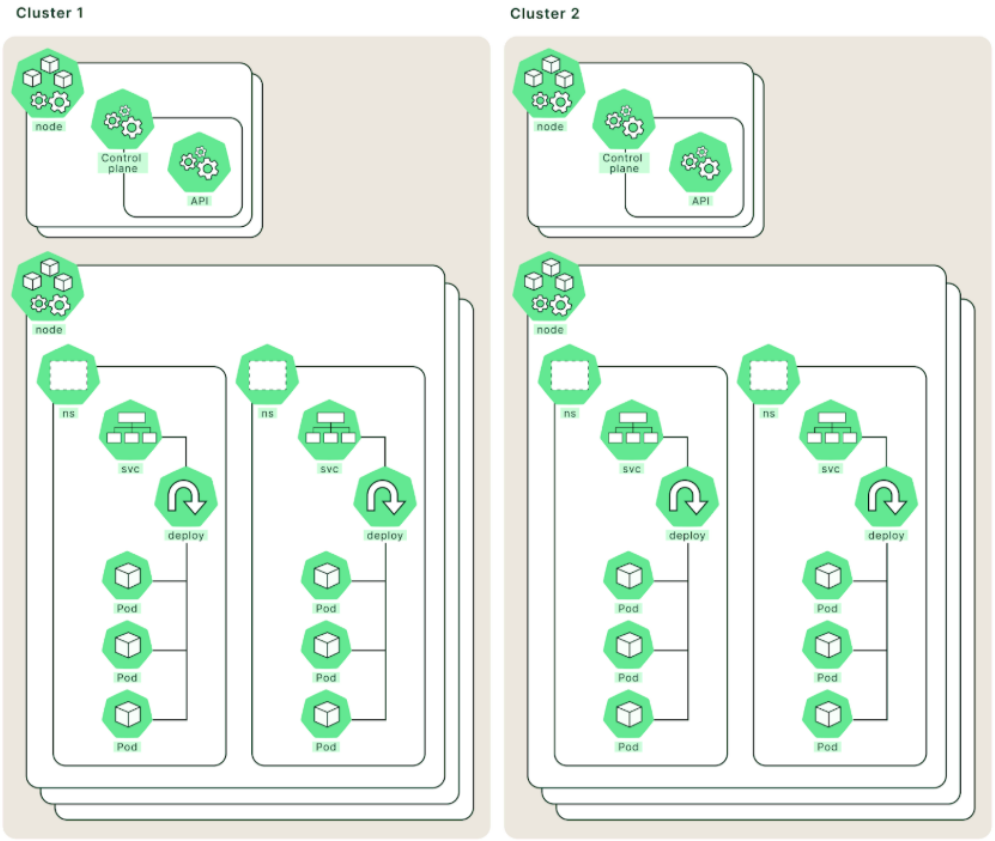
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/workloads/pods/. For example, components of the first cluster architecture differ from the second cluster architecture based on worker node configurations, the running containerized applications within Pods, networking, and other features part of Kubernetes.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Cluster Architecture The architectural concepts behind Kubernetes. A Kubernetes cluster consists of a control plane plus a set of worker machines, called nodes, that run containerized applications. Every cluster needs at least one worker node in order to run Pods. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability. Source: https://kubernetes.io/docs/concepts/architecture/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://www.kubecost.com/kubernetes-multi-cloud/kubernetes-multi-cluster/. As a further example, multiple cluster configurations can differ based on different environmental access, location of the cluster, performance, and other considerations as mentioned below.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**

	There are many reasons to want to run multiple clusters, including but not limited to:
	• Location
	  • Latency: it can be necessary to deploy the application as close to the customers as possible.
	  • Jurisdiction: it can be mandated to keep user data in-country.
	  • Data gravity: data already exists in one provider, but it can be decided to run the application in another environment.
	• Isolation
	  • Environment (e.g. dev, test, prod)
	  • Performance isolation: a workload may consume too many resources, at the expense of other workloads.
	  • Security isolation: sensitive data or untrusted code must be isolated in their own environments.
	  • Organizational isolation: teams may have different management domains.
	  • Cost isolation: multitenancy can greatly complexify billing management for different teams.
	• Reliability
	  • Blast radius: an infrastructure or application incident in one cluster must not impact the whole system.
	  • Infrastructure diversity: an underlying zone, region, or provider outage does not bring down the whole system.
	  • Scale: the application is too big to fit in a single cluster.
	  • Upgrade scope: some parts of the application may require an infrastructure upgrade, that may impact other parts of the application. Having multiple clusters can also avoid the need for in-place cluster upgrades.
	Source: https://multicluster.sigs.k8s.io/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	For example, Kubernetes implements network policies for controlling the traffic at the IP address or port level within the cluster, and between Pods and the outside world. Network policies limit access by only allowing specific IP ranges to request. **Network Policies** If you want to control traffic flow at the IP address or port level (OSI layer 3 or 4), NetworkPolicies allow you to specify rules for traffic flow within your cluster, and also between Pods and the outside world. Your cluster must use a network plugin that supports NetworkPolicy enforcement. Source: https://kubernetes.io/docs/concepts/services-networking/network-policies/. The entities that a Pod can communicate with are identified through a combination of the following three identifiers: 1. Other pods that are allowed (exception: a pod cannot block access to itself) 2. Namespaces that are allowed 3. IP blocks (exception: traffic to and from the node where a Pod is running is always allowed, regardless of the IP address of the Pod or the node) Source: https://kubernetes.io/docs/concepts/services-networking/network-policies/.

[redacted]

service/networking/networkpolicy.yaml

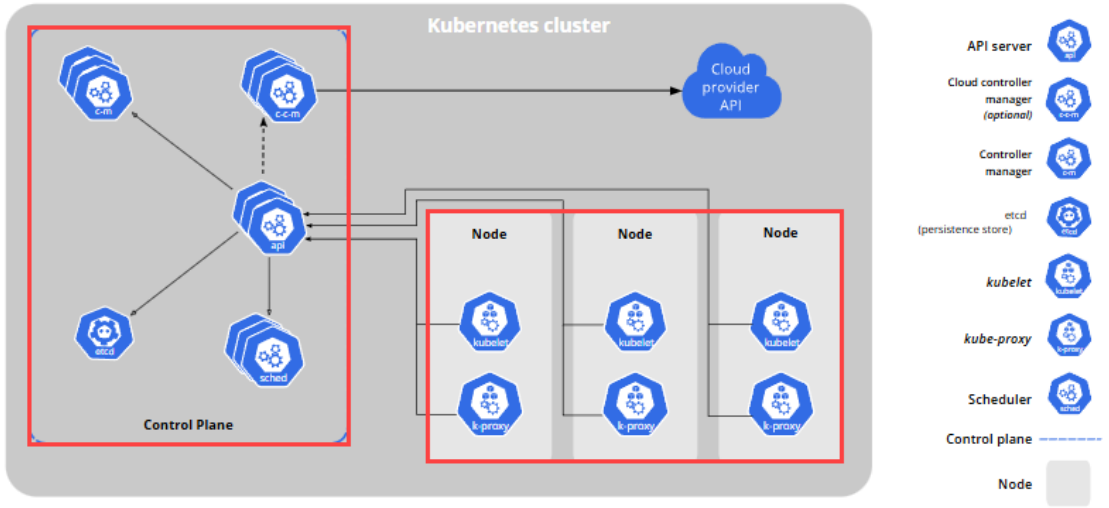
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

Source: https://kubernetes.io/docs/concepts/services-networking/network-policies/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	So, the example NetworkPolicy: 1. isolates `role=db` pods in the `default` namespace for both ingress and egress traffic (if they weren't already isolated) 2. (Ingress rules) allows connections to all pods in the `default` namespace with the label `role=db` on TCP port 6379 from: ◦ any pod in the `default` namespace with the label `role=frontend` ◦ any pod in a namespace with the label `project=myproject` ◦ IP addresses in the ranges `172.17.0.0` – `172.17.0.255` and `172.17.2.0` – `172.17.255.255` (ie, all of `172.17.0.0/16` except `172.17.1.0/24`) 3. (Egress rules) allows connections from any pod in the `default` namespace with the label `role=db` to CIDR `10.0.0.0/24` on TCP port 5978 Source: https://kubernetes.io/docs/concepts/services-networking/network-policies/.
[1.g] wherein the computing resources comprise processing nodes, data storage shared by the processing nodes, and at least one communications network to link the processing nodes	On information and belief, the Southwest Count IX Systems and Services include a system where the computing resources comprise processing nodes, data storage shared by the processing nodes, and at least one communications network to link the processing nodes to each other and to the data storage. For example, organized groups of resources such as CPU and memory are accessed by nodes of Kubernetes worker machines. Nodes are managed by the Kubernetes control plane, where a pod and its containers run on a node.

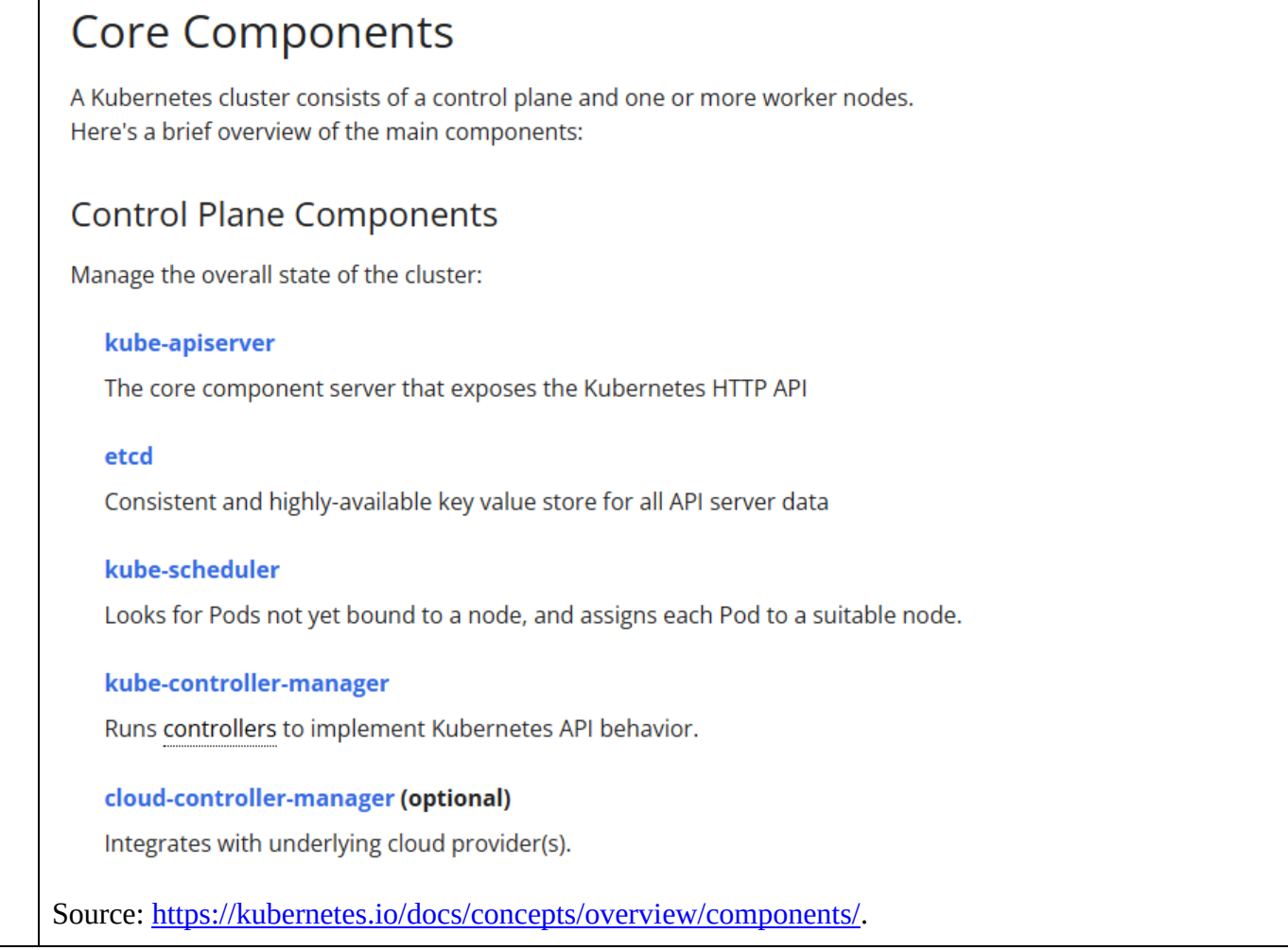
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
to each other and to the data storage;	 The components of a Kubernetes cluster Source: https://kubernetes.io/docs/concepts/overview/components/.

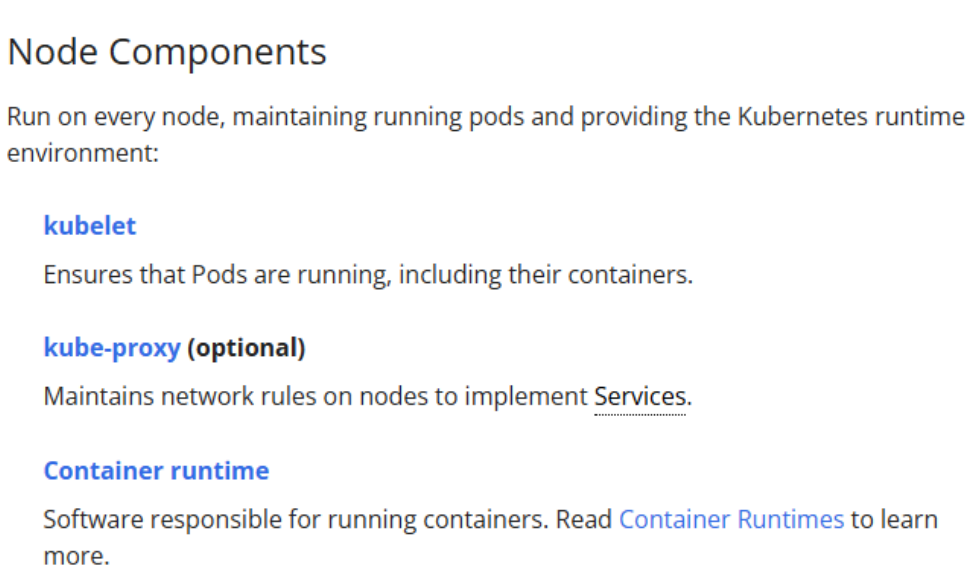
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Resource Management for Pods and Containers When you specify a Pod, you can optionally specify how much of each resource a container needs. The most common resources to specify are CPU and memory (RAM); there are others. When you specify the resource *request* for containers in a Pod, the kube-scheduler uses this information to decide which node to place the Pod on. When you specify a resource *limit* for a container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set. The kubelet also reserves at least the *request* amount of that system resource specifically for that container to use. Source: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/.

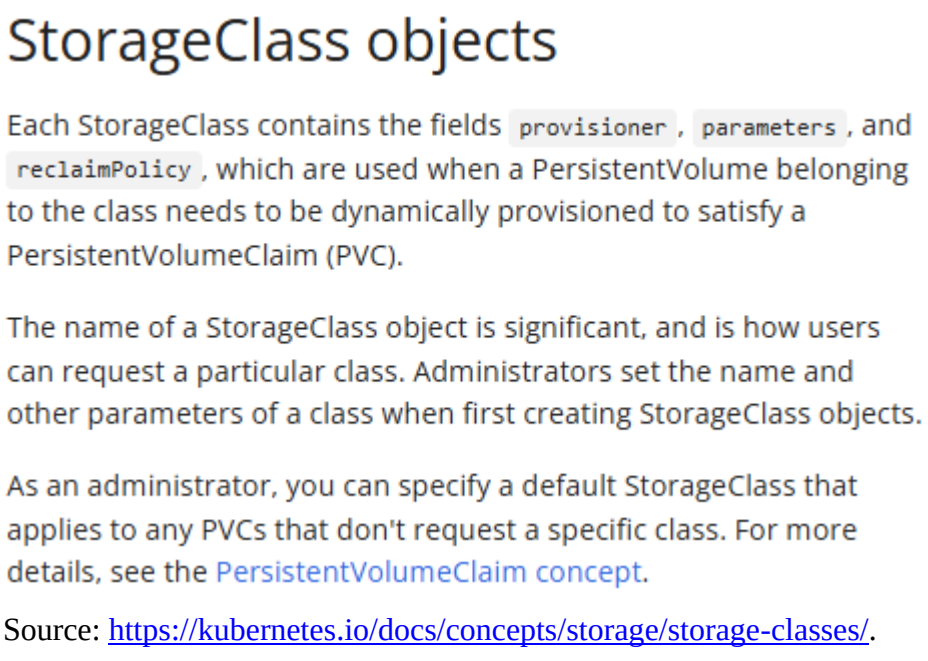
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/overview/components/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/concepts/overview/components/. Furthermore, Kubernetes includes centralized or distributed storage shared across nodes in a cluster. For example, this includes Persistent Volumes, which are provisioned by an administrator for storage across Pods and/or containers across all nodes within a Kubernetes cluster. A *PersistentVolume* (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes, but have a lifecycle independent of any individual Pod that uses the PV. This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

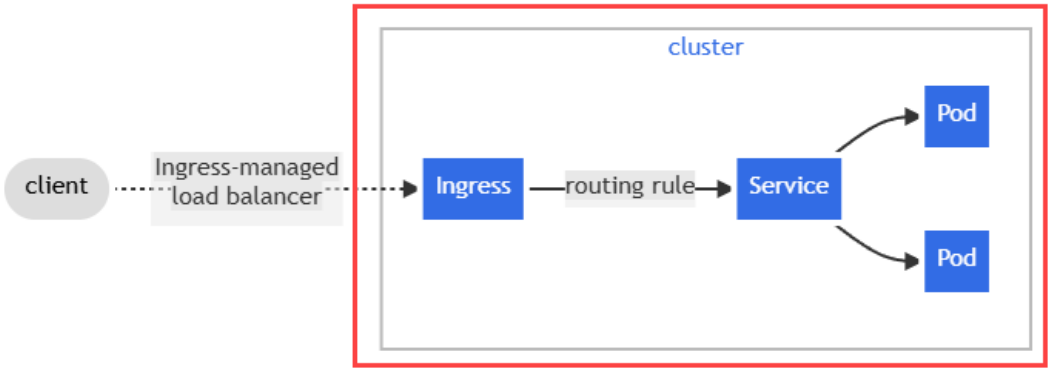
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Source: https://kubernetes.io/docs/concepts/storage/persistent-volumes/.  Source: https://kubernetes.io/docs/concepts/storage/storage-classes/.
[1.h] wherein the first cluster establishes communications between the set of computing resources of the first cluster and a first gateway communicatively linked between the first cluster and the private communications network;	On information and belief, the Southwest Count IX Systems and Services include a system where the first cluster establishes communications between the set of computing resources of the first cluster and a first gateway communicatively linked between the first cluster and the private communications network. For example, Kubernetes clusters include at least one node, where a node is either a physical or virtual machine comprising a CPU or portion of CPU resources and memory, to run workloads. These clusters are connected to a private network, for example, the Southwest accused network via a gateway, for example, such as Ingress controllers, internal load balancers, and cloud private endpoints. On information and belief, the private network, for example, the Southwest accused network, facilitates container-to-container, Pod-to-Pod, and/or Pod-to-Services communications across multiple clusters for sharing data and handling tasks.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Ingress Make your HTTP (or HTTPS) network service available using a protocol-aware configuration mechanism, that understands web concepts like URIs, hostnames, paths, and more. The Ingress concept lets you map traffic to different backends based on rules you define via the Kubernetes API. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/. **Terminology** For clarity, this guide defines the following terms: • Node: A worker machine in Kubernetes, part of a cluster. • Cluster: A set of Nodes that run containerized applications managed by Kubernetes. For this example, and in most common Kubernetes deployments, nodes in the cluster are not part of the public internet. • Edge router: A router that enforces the firewall policy for your cluster. This could be a gateway managed by a cloud provider or a physical piece of hardware. • Cluster network: A set of links, logical or physical, that facilitate communication within a cluster according to the Kubernetes networking model. • Service: A Kubernetes Service that identifies a set of Pods using label selectors. Unless mentioned otherwise, Services are assumed to have virtual IPs only routable within the cluster network. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	What is Ingress? Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. Here is a simple example where an Ingress sends all its traffic to one Service:  Figure. Ingress Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.
[1.i] wherein the second cluster establishes communications among the set of computing resources of the second cluster and a second gateway communicatively linked between the second cluster and the private communications network;	On information and belief, the Southwest Count IX Systems and Services include a system where the second cluster establishes communications among the set of computing resources of the second cluster and a second gateway communicatively linked between the second cluster and the private communications network. Kubernetes clusters include at least one node, where a node is either a physical or virtual machine comprising a CPU or portion of CPU resources and memory, to run workloads. On information and belief, in Kubernetes, Services provide a way for communication within a second cluster to the private network, for example, the Southwest accused network via a gateway, such as Ingress controllers, internal load balancers, and cloud private endpoints. On information and belief, the

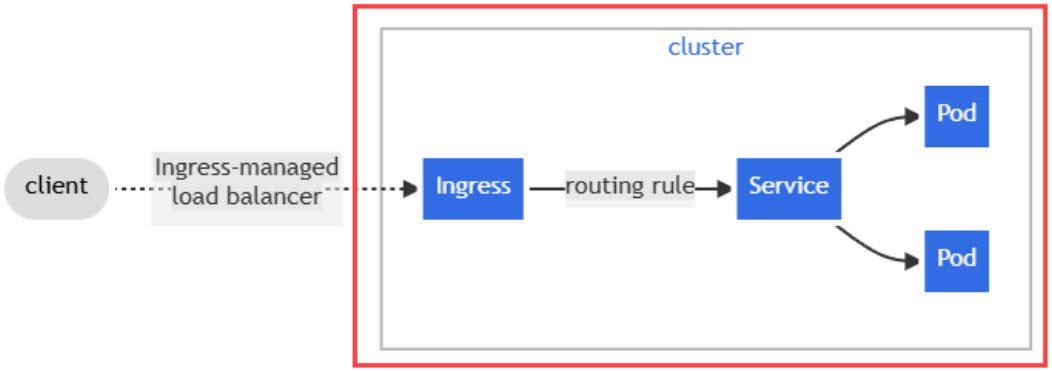
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	private network, for example, the Southwest accused network, facilitates container-to-container, Pod-to-Pod, and/or Pod-to-Services communications across multiple clusters for sharing data and handling tasks. **Ingress** Make your HTTP (or HTTPS) network service available using a protocol-aware configuration mechanism, that understands web concepts like URIs, hostnames, paths, and more. The Ingress concept lets you map traffic to different backends based on rules you define via the Kubernetes API. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Terminology For clarity, this guide defines the following terms: • Node: A worker machine in Kubernetes, part of a cluster. • Cluster: A set of Nodes that run containerized applications managed by Kubernetes. For this example, and in most common Kubernetes deployments, nodes in the cluster are not part of the public internet. • Edge router: A router that enforces the firewall policy for your cluster. This could be a gateway managed by a cloud provider or a physical piece of hardware. • Cluster network: A set of links, logical or physical, that facilitate communication within a cluster according to the Kubernetes networking model. • Service: A Kubernetes Service that identifies a set of Pods using label selectors. Unless mentioned otherwise, Services are assumed to have virtual IPs only routable within the cluster network. Source: https://kubernetes.io/docs/concepts/services-networking/ingress/.

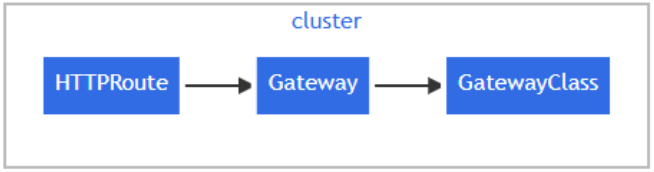
[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	What is Ingress? Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource. Here is a simple example where an Ingress sends all its traffic to one Service:  Figure. Ingress Source: https://kubernetes.io/docs/concepts/services-networking/ingress/. A Kubernetes cluster network includes the Gateway API, which provides Gateway resources. These resources can be defined and specified so that external traffic from the Internet can be routed to various Services.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	Gateway API Gateway API is a family of API kinds that provide dynamic infrastructure provisioning and advanced traffic routing. Make network services available by using an extensible, role-oriented, protocol-aware configuration mechanism. Gateway API is an add-on containing API kinds that provide dynamic infrastructure provisioning and advanced traffic routing. Source: https://kubernetes.io/docs/concepts/services-networking/gateway/. • The Gateway API (or its predecessor, Ingress) allows you to make Services accessible to clients that are outside the cluster. Source: https://kubernetes.io/docs/concepts/services-networking/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	# Resource model Gateway API has three stable API kinds: • **GatewayClass:** Defines a set of gateways with common configuration and managed by a controller that implements the class. • **Gateway:** Defines an instance of traffic handling infrastructure, such as cloud load balancer. • **HTTPRoute:** Defines HTTP-specific rules for mapping traffic from a Gateway listener to a representation of backend network endpoints. These endpoints are often represented as a Service. Gateway API is organized into different API kinds that have interdependent relationships to support the role-oriented nature of organizations. A Gateway object is associated with exactly one GatewayClass; the GatewayClass describes the gateway controller responsible for managing Gateways of this class. One or more route kinds such as HTTPRoute, are then associated to Gateways. A Gateway can filter the routes that may be attached to its `listeners`, forming a bidirectional trust model with routes. The following figure illustrates the relationships of the three stable Gateway API kinds:  Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**

Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

Design principles

The following principles shaped the design and architecture of Gateway API:

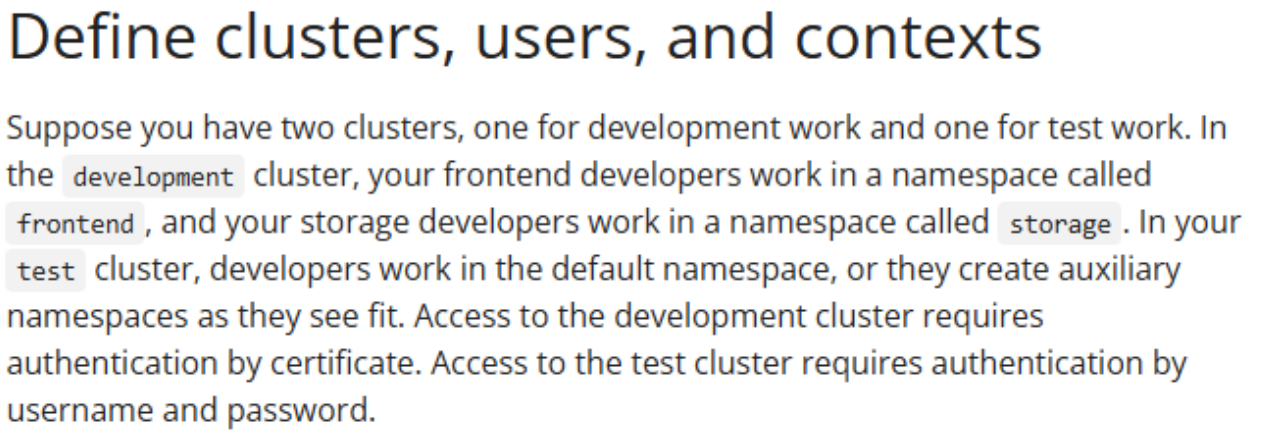
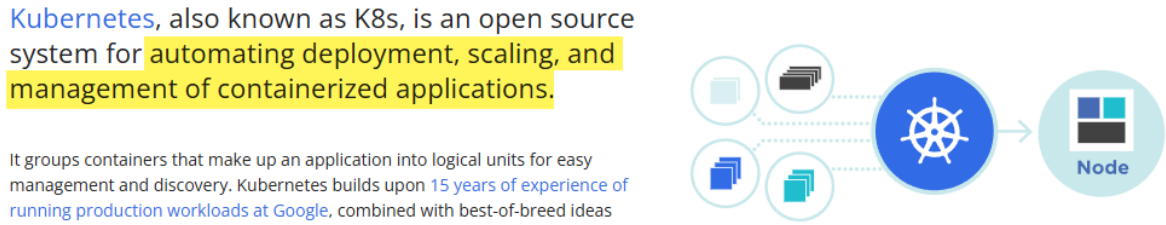
- **Role-oriented:** Gateway API kinds are modeled after organizational roles that are responsible for managing Kubernetes service networking:
  - **Infrastructure Provider:** Manages infrastructure that allows multiple isolated clusters to serve multiple tenants, e.g. a cloud provider.
  - **Cluster Operator:** Manages clusters and is typically concerned with policies, network access, application permissions, etc.
  - **Application Developer:** Manages an application running in a cluster and is typically concerned with application-level configuration and Service composition.
- **Portable:** Gateway API specifications are defined as custom resources and are supported by many implementations.
- **Expressive:** Gateway API kinds support functionality for common traffic routing use cases such as header-based matching, traffic weighting, and others that were only possible in Ingress by using custom annotations.
- **Extensible:** Gateway allows for custom resources to be linked at various layers of the API. This makes granular customization possible at the appropriate places within the API structure.

Source: https://kubernetes.io/docs/concepts/services-networking/gateway/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
[1.j] wherein communications between the first cluster and the second cluster are isolated;	On information and belief, the Southwest Count IX Systems and Services include a system where communications between the first cluster and the second cluster are isolated. For example, multi-cluster Kubernetes provides for isolated workloads in separate clusters. **Workload Isolation** Multi-cluster Kubernetes allows you to isolate different workloads, such as development, staging, and production environments, in separate clusters. This improves fault tolerance and reduces the risk of cascading failures. Workload isolation also enables you to enforce strict resource quotas and security policies on a per-cluster basis. Source: https://www.tigera.io/learn/guides/kubernetes-networking/kubernetes-multi-cluster/. Multi-cluster Kubernetes also provides for configuring access to different clusters. **Configure Access to Multiple Clusters** This page shows how to configure access to multiple clusters by using configuration files. After your clusters, users, and contexts are defined in one or more configuration files, you can quickly switch between clusters by using the `kubectl config use-context` command.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Source: https://kubernetes.io/docs/tasks/access-application-cluster/configure-access-multiple-clusters/.
[1.k] wherein the first cluster is a high performance cluster; and	On information and belief, the Southwest Count IX Systems and Services include a system where the first cluster is a high-performance cluster. For example, Kubernetes is a system for automating deployment, scaling and management or containerized applications.  Source: https://kubernetes.io/.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	 Planet Scale Designed on the same principles that allow Google to run billions of containers a week, Kubernetes can scale without increasing your operations team. Source: https://kubernetes.io/. Kubernetes is a scalable and performant engine that orchestrates containers in a server environment. It is highly optimized by default, and it scales nicely in a suitable infrastructure. It is also less opinionated by default, and there are plenty of customizations for end-users to define. This flexibility allows Kubernetes to cover many different use cases and penetrate the market faster, making it extremely popular. Source: https://platform9.com/blog/10-kubernetes-performance-tips/. As a non-limiting example, HPC workloads include many tasks, servers, and parallelization of processing deployed via Kubernetes.

[redacted]

U.S. Patent No. 8,352,584 (Claim 1)	
Claim 1	**Example Southwest Count IX Systems and Services**
	In Kubernetes, the base unit of scheduling is a Pod: one or more Docker containers scheduled to a cluster host. Kubernetes assumes that workloads are containers. While Kubernetes has the notion of Cron Jobs and Jobs that run to completion, applications deployed on Kubernetes are typically long-running services, like web servers, load balancers or data stores and while they are highly dynamic with pods coming and going, they differ greatly from HPC application patterns. Traditional HPC applications often exhibit different characteristics: • In financial or engineering simulations, a job may be comprised of tens of thousands of short-running tasks, demanding low-latency and high-throughput scheduling to complete a simulation in an acceptable amount of time. • A computational fluid dynamics (CFD) problem may execute in parallel across many hundred or even thousands of nodes using a message passing library to synchronize state. This requires specialized scheduling and job management features to allocate and launch such jobs and then to checkpoint, suspend/resume or backfill them. • Other HPC workloads may require specialized resources like GPUs or require access to limited software licenses. Organizations may enforce policies around what types of resources can be used by whom to ensure projects are adequately resourced and deadlines are met. Source: https://kubernetes.io/blog/2017/08/kubernetes-meets-high-performance/#:~:text=HPC%20workloads%20unique%20challenges.

[redacted]

[1.l] wherein the second cluster is a high performance cluster.	On information and belief, the Southwest Count IX Systems and Services include a system where the second cluster is a high performance cluster. For example, Kubernetes is a system for automating deployment, scaling and management or containerized applications. Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.  Source: https://kubernetes.io/.  Planet Scale Designed on the same principles that allow Google to run billions of containers a week, Kubernetes can scale without increasing your operations team. Source: https://kubernetes.io/. Kubernetes is a scalable and performant engine that orchestrates containers in a server environment. It is highly optimized by default, and it scales nicely in a suitable infrastructure. It is also less opinionated by default, and there are plenty of customizations for end-users to define. This flexibility allows Kubernetes to cover many different use cases and penetrate the market faster, making it extremely popular. Source: https://platform9.com/blog/10-kubernetes-performance-tips/.

[redacted]

	As a non-limiting example, HPC workloads include many tasks, servers, and parallelization of processing deployed via Kubernetes. In Kubernetes, the base unit of scheduling is a Pod: one or more Docker containers scheduled to a cluster host. Kubernetes assumes that workloads are containers. While Kubernetes has the notion of Cron Jobs and Jobs that run to completion, applications deployed on Kubernetes are typically long-running services, like web servers, load balancers or data stores and while they are highly dynamic with pods coming and going, they differ greatly from HPC application patterns. Traditional HPC applications often exhibit different characteristics: • In financial or engineering simulations, a job may be comprised of tens of thousands of short-running tasks, demanding low-latency and high-throughput scheduling to complete a simulation in an acceptable amount of time. • A computational fluid dynamics (CFD) problem may execute in parallel across many hundred or even thousands of nodes using a message passing library to synchronize state. This requires specialized scheduling and job management features to allocate and launch such jobs and then to checkpoint, suspend/resume or backfill them. • Other HPC workloads may require specialized resources like GPUs or require access to limited software licenses. Organizations may enforce policies around what types of resources can be used by whom to ensure projects are adequately resourced and deadlines are met. Source: https://kubernetes.io/blog/2017/08/kubernetes-meets-high-performance/#:~:text=HPC%20workloads%20unique%20challenges.